Threat Modeling Using Trike

Methodology Overview

Brenda Larcom
February 27, 2012

http://www.octotrike.org/
Agenda

What are we going to talk about?

• Differences
• Theory
• Anatomy
• Creation
• Use
• Tool Support
Differences

What’s so cool about Trike?

• Generate threats [semi]-automatically, no brainstorming
• Security-inexperienced developers reliably find issues
• Security geeks can pick up where developers left off
• It’s clear what to analyze
• It’s clear when to stop
• Attack chaining, not attack trees
• Tools provide immediate feedback as you design
• Start earlier, with requirements
• Include sequences of events, not just static architecture
• Include intended system behavior
Differences

What’s the catch?

• Tools & methodology hard-code theory
• Heavily reliant on automation
• All available tools are bleeding edge
• Requires more data about the system
• Different, more restrictive definitions
• It’s clear when you’ve stopped too soon
Theory

What are Trike’s basic assumptions?

• Models are for answering questions
• Threat models can only answer technical questions
• Developers know about the system, security geeks know about security
  • The reverse may not be true
• Secure enough = meets security objectives
• Attacker goals are irrelevant
• Threats = \( f(\text{system}) \)
• Attackers will use both intended & unintended system behavior
Anatomy

What goes into a threat model?

- Functional requirements
  - Actors
  - Assets
  - Intended actions
- Deployment environment
- Security objectives
  - In-scope requirements threats
  - In-scope attackers
- System architecture
  - Static view
  - Dynamic view
  - Security attributes & technology

- Requirements threats
- Architecture threats
  - Feasibility
- Relationships between threats
  - Impact
- Mitigations
  - Effectiveness
Creation

What do I do?
Creation

What happens at requirements time?

- Intended actions, per business stakeholders
- Automatically generated threats
- Threat impact, per business stakeholders
- Security objectives, per all stakeholders

<table>
<thead>
<tr>
<th>#</th>
<th>Initial Configuration</th>
<th>Attackers</th>
<th>Prohibited Threats</th>
<th>Intended Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>SO1</td>
<td>Blog exists</td>
<td>Anonymous</td>
<td>Create Blog Entry on someone else's Blog</td>
<td>Thwart</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User</td>
<td>Delete Blog Entry</td>
<td>Detect and Log</td>
</tr>
<tr>
<td></td>
<td>Blog is not configured to allow posting by the attacking user</td>
<td>User</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creation

What do/did I do?

Prune analysis using security objectives
Creation

What happens at architecture time?

- Privilege analysis
  - Finds inconsistencies and issues in privileges a component or connection has, provides, revokes, uses, and requires
  - Likely automatable; theory still in development
- HAZOP analysis
  - Structured analysis technique from safety engineering
  - Identifies harmful variations in sequences of events
  - Semi-automatable
- Attack chaining
  - Collections of privileges are nodes, intended and unintended behaviors are edges
  - Definitely automatable
  - Need to investigate scaling/computational complexity issues
  - Prototype in development

You can do this now
Creation

What does a sequence of events look like?

<table>
<thead>
<tr>
<th>Use Case #</th>
<th>Step #</th>
<th>Choice</th>
<th>Choice</th>
<th>Terminal</th>
<th>Actor</th>
<th>Action</th>
<th>Object</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>User</td>
<td>submits</td>
<td>Blog Entry</td>
<td>to Web Server</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>AV Filter</td>
<td>scans</td>
<td>Blog Entry</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Virus</td>
<td></td>
<td></td>
<td>Web Server</td>
<td>sends</td>
<td>Status Response</td>
<td>to User reflecting failure due to virus</td>
<td></td>
</tr>
<tr>
<td></td>
<td>No Virus</td>
<td></td>
<td></td>
<td>Blog Module</td>
<td>compares</td>
<td>User Account</td>
<td>to Blog Permissions</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>No Virus</td>
<td>Authorized</td>
<td></td>
<td>Blog Module</td>
<td>creates</td>
<td>Blog Entry</td>
<td>in Database</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>No Virus</td>
<td>Authorized</td>
<td></td>
<td>Blog Module</td>
<td>creates</td>
<td>Log Entry</td>
<td>in Database</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>No Virus</td>
<td>Authorized</td>
<td></td>
<td>Blog Module</td>
<td>sends</td>
<td>Status Response</td>
<td>to User reflecting success</td>
<td></td>
</tr>
</tbody>
</table>

STACH&LIU
## Creation

### How do I vary a step?

<table>
<thead>
<tr>
<th>Use Case #</th>
<th>Step #</th>
<th>Condition</th>
<th>Varied Element</th>
<th>Guide Word</th>
<th>Guide Word Meaning</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC3</td>
<td>1</td>
<td>to Server</td>
<td>Actor</td>
<td>NO</td>
<td>Actor is not in the correct role, or does not have the capability.</td>
<td>A User who is not logged in submits a Blog Entry to the Web Server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Actor</td>
<td>AS WELL AS</td>
<td>Actor is in the correct role, but is also in another (typically more privileged) role or otherwise has additional capabilities.</td>
<td>A User who is also an Admin submits a Blog Entry to the Web Server.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Actor</td>
<td>PART OF</td>
<td>Actor has some, but not all of the needed capabilities.</td>
<td>The attacker submits a Blog Entry to the Web Server from a public terminal a User just logged out of.</td>
</tr>
</tbody>
</table>
## Creation

### How do I analyze a variation?

<table>
<thead>
<tr>
<th>Use Case #</th>
<th>Step #</th>
<th>Variation</th>
<th>Security Objectives Variation Would Help Attacker Achieve</th>
<th>Rationale for Variation’s Helpfulness to Attacker</th>
<th>Attacker Influenced</th>
<th>Rationale for Attacker Influenced</th>
<th>Issue Criticality</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC3</td>
<td>1</td>
<td>A User who is not logged in submits a Blog Entry to the Web Server.</td>
<td></td>
<td>There is no step that checks whether the User is actually logged in; the Web Server will accept any Blog Entry sent to this interface.</td>
<td>Red</td>
<td>There’s nothing stopping anyone on the Internet from submitting a Blog Entry.</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A User who is also an Admin submits a Blog Entry to the Web Server.</td>
<td></td>
<td>Our security objectives trust all Admins.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The attacker submits a Blog Entry to the Web Server from a public terminal a User just logged out of.</td>
<td></td>
<td>On logout, the server invalidates the User’s session and instructs the client to delete all cookies; a terminal the User logged out of is no more useful than a terminal the User has never used.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How do I use a threat model to make decisions?

- Identify a project decision that should be affected by security
  - E.g. Whether application is ready to launch
- Identify information that should inform that decision
  - E.g. Does the expense reports application meet its security objectives?
- Extract that information from the model
  - E.g. Examine threats that are still feasible for unbroken chains from attacker starting privileges to prohibited threats
Use

How do I use a threat model at design time?

- Security objectives should be met
- Defenses should be protecting against threats
- Apply design patterns appropriately to respond to threats (e.g. input trust boundary, centralized input validation library)
- Best design has either fewer or easier threats to defend against
Use

How do I use a threat model to drive security tests?

- Confirm protections are in place
- Confirm responsibilities are met
- Try to perform all the relevant threats identified in the threat model
  - Start with those that are more beneficial to the attacker
Tool Support

What can I have Right Now?

• **Trike 1, in Squeak**
  - Auto-generates threats based on intended actions & lets you prioritize them
  - Auto-generates attack tree stubs (deprecated)
  - No file import or export

• **Trike 1.5, as a spreadsheet**
  - Auto-generates threats based on intended actions and deployment environment & lets you prioritize them
  - Security objectives
  - Data collection, but no analysis (yet) for component & connection privileges
  - Data collection & basic support for HAZOP analysis
  - Updated regularly

http://www.octotrike.org/
Tool Support

Where is this headed?

• Trike 2, in Squeak
  • Have some code, in re-design now
  • Will implement everything discussed here
  • Sketch-based interface that highlights problems and missing information as you draw
  • REST interface in case you hate our futuristic UI enough to write a different one
  • Yes, it will do files, I promise
  • No firm ETA yet, but 2013 is more likely than 2012
  • Security objectives portion will likely come out first
Thanks

- Eleanor Saitta
- Erik Simmons
- Khyati Shrivastava

- Mozilla!

For more information, see [http://www.octotrike.org/](http://www.octotrike.org/).